NUI Galway
OÉ Gaillimh

Autumn Examinations 2013

Exam Code(s)	4IF, 4BP, 3BA, 1SD
Exam(s)	4th Year B.Sc. (Information Technology) 4th Year B.E. (Electronic & Computer Engineering) 3rd Year B.A. (Information Technology) Higher Diploma in Applied Science (Software Design & Development)
Module Code(s)	CT404, CT336
Module(s)	GRAPHICS AND IMAGE PROCESSING
Paper No.	I
Repeat Paper	
External Examiner(s)	Prof. M. O'Boyle
Internal Examiner(s)	Prof. G. Lyons Dr. M. Madden * Dr. S. Redfern
Instructions:	Answer any three questions. All questions carry equal marks.
Duration	2 hours
No. of Pages	6
Department(s)	Information Technology
Course Co-ordinator(s)	

Requirements:

MCQ	Release to Library: Yes ☐ No ☐
Handout	
Statistical/ Log Tables	
Cambridge Tables	
Graph Paper	Required
Log Graph Paper	
Other Materials	

Q1.

Consider the OpenGL program below, which draws a 2-dimensional star shape from two triangles (see Fig. 1).

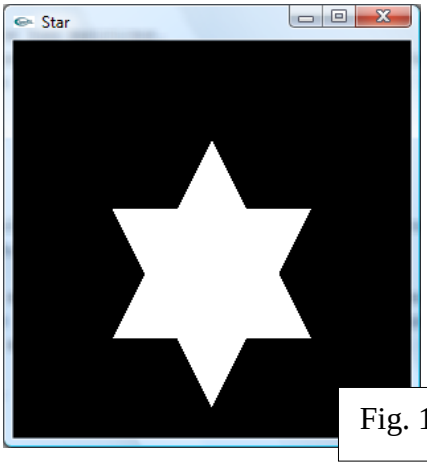


Fig. 1

```
#include <gl/glut.h>

void drawStar() {
  glBegin(GL_TRIANGLES);
    glVertex2f(-0.5, -0.5);
    glVertex2f(0.5, -0.5);
    glVertex2f(0.0, 0.5);

    glVertex2f(-0.5, 0.15);
    glVertex2f(0.5, 0.15);
    glVertex2f(0.0, -0.85);
  glEnd();
}

void display(void) {
  glClear (GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

  glMatrixMode(GL_MODELVIEW);
  glLoadIdentity();
  gluOrtho2D(-1, 1, -1, 1);

  drawStar();

  glFlush ();
}

int main(int argc, char** argv) {
   glutInit(&argc, argv);
   glutCreateWindow ("Star");
   glutDisplayFunc(display);
   glutMainLoop();
   return 0;
}
```

Fig. 2

a) Modify the display() function so that a 3-dimensional (perspective) projection is used rather than an orthographic projection [2 marks]

b) Modify the display() and drawStar() functions so that the star is drawn at a 3-dimensional location and colour as specified in its parameter list [6 marks]

```
drawStar(float x, float y, float z, float r, float g, float b)
```

c) Extend the program so that a 'starfield' animation is displayed using the Idle Callback approach. Multiple stars should be stored in an array, and initialised to random x, y and z co-ordinates. They should move towards the camera along the z axis, and when they pass the camera they should be moved back to the far distance again. The final program should produce a result similar to that shown in Fig. 2. [12 marks]

Note that some useful OpenGL functions, and other standard general-purpose C functions, are listed at the end of this exam paper.

Q2.

(a) Describe the use of extrusion in X3D, referring to each of the seven fields used by the Extrusion node. **Note that extrusion and other useful nodes from the X3D language are summarised on the final page of this exam paper**. [5 marks]

(b) Write X3D code to make a model of a free-standing room lamp (example illustrated on the right, Fig. 3).

- Include a diagram of your model showing its measurements, and make the model as geometrically accurate as possible [9 marks]
- The lamp should be constructed from a shiny white material [3 marks]
- A TouchSensor should be used to turn a PointLight inside the head of the lamp on and off [3 marks]

Fig. 3

Q3.

a) In 3D computer graphics, what is a surface normal? Illustrate your answer with a diagram. [3 marks]

b) Discuss, with the use of a diagram in each case, the following 3D graphics techniques. In each case, identify the importance of surface normals:

i. Flat (Lambert) Shading
ii. Normal Interpolating (Phong) Shading
iii. Radiosity
iv. Binary Space Partitioning [8 marks]

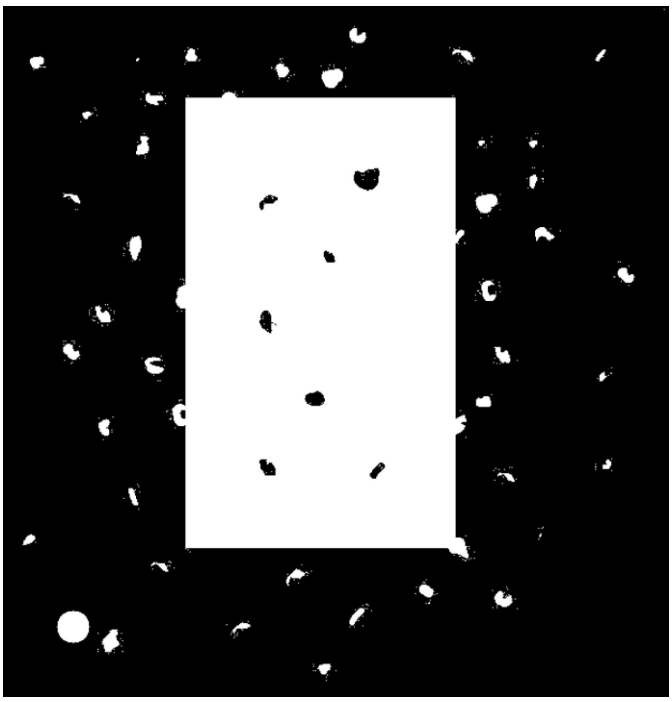
c) Explain the keyframe approach to animation in computer graphics, and explain its use in X3D, referring to the TimeSensor, Transform, OrientationInterpolator and PositionInterpolator nodes in your answer. [4 marks]

d) Rather than using hard-coded animation parameters, a more powerful approach to producing animations in X3D is to use JavaScript nodes to dynamically calculate key positions or orientations. Write X3D code for a JavaScript node which produces a smooth elliptical motion, and which is suitable for use with a TimeSensor node and a Transform node. [5 marks]

Q4.

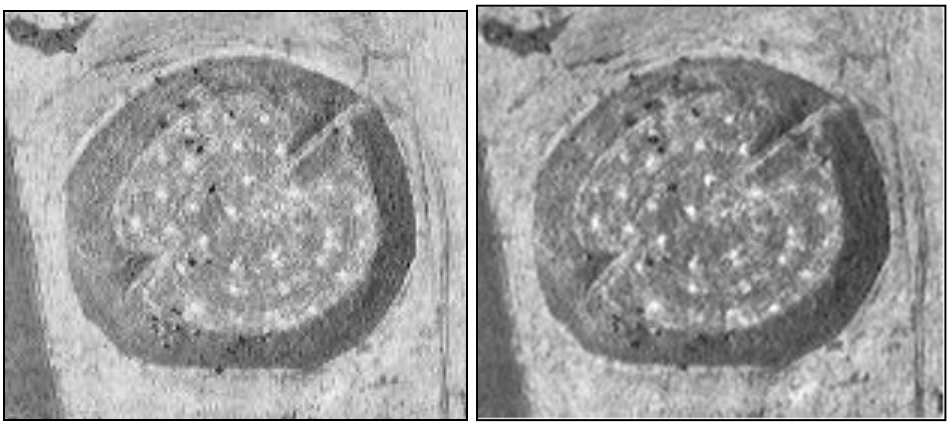
a) Describe the *mathematical morphology* approach to image processing. Outline some typical circumstances in which this approach is useful. [4 marks]

b) The image below contains a large white rectangle and a number of patches of noise. Outline and defend a morphology-based algorithm for automatic isolation of the rectangle. Use sketches to indicate approximately how the image would appear following each step of your solution. [7 marks]



c) With regard to the use of image processing for the capturing of 3 Dimensional (3D) information, define the terms: (i) active depth sensing, (ii) passive depth sensing. [3 marks]

d) Outline, at a high level, a suitable algorithm for extracting 3D information from a stereo pair of images, such as the image pair illustrated below. [6 marks]



Q5.

a) Edge Detection is a basic image processing technique, which is often used as one of the earlier steps in image processing systems. Discuss the role of edge detection in a typical object extraction task. Why would the image resulting from edge detection be useful to the later image processing steps in your task? Be specific in your answer, choosing at least one image processing algorithm to use after edge detection, explaining why an edge-enhanced image is useful to this algorithm.

[6 marks]

b) Outline the following algorithms, and discuss their relative strengths and weaknesses: (i) convolution-based edge detection, (ii) compass edge detection, (iii) the 'Canny' algorithm. Provide diagrams in each case.

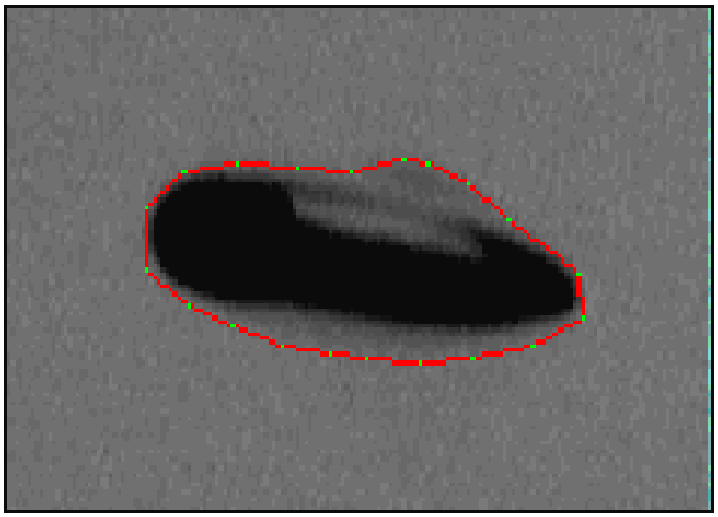
[6 marks]

c) Outline and discuss the image processing technique called 'active contours'.

[4 marks]

d) Present a suitable set of optimisation constraints (sometimes called energy factors) for accurately tracing the outline of a bubble such as the one shown below, using active contours.

[4 marks]



Some useful X3D nodes:

Node	Important Fields and Nested Nodes
Shape	Nested Nodes: Appearance, Geometry Nodes (Box, Sphere, Cone, Cylinder, Text, Extrusion, etc.)
Appearance	Nested Nodes: Material, ImageTexture
Material	Fields: diffuseColor, specularColor, emissiveColor, ambientIntensity, transparency, shininess
ImageTexture	Fields: url
Transform	Fields: translation, rotation, scale, center. Nested Nodes: Other Transforms, Shapes, Sensors
TimeSensor	Fields: enabled, startTime, stopTime, cycleInterval, loop
PositionInterpolator	Fields: key, keyValue
OrientationInterpolator	Fields: key, keyValue
Extrusion	Fields: crossSection, spine, scale, orientation, beginCap, endCap, creaseAngle
Box	Fields: size
Sphere	Fields: radius
Cylinder	Fields: radius, height, side, top, bottom
Cone	Fields: height, bottomRadius, side, bottom
PointLight	Fields: on, location, radius, intensity, ambientIntensity, color, attenuation
ROUTE	Fields: fromNode, fromField, toNode, toField

Some useful OpenGL functions:

Function	Comments
glBegin(type), glEnd()	Together these delimit a set of vertices. 'Type' defines how to interpret them (GL_POLYGON, GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN, GL_QUADS)
glVertex2f(x,y), glVertex3f(x,y,z)	Define a vertex
glColor3f(r,g,b)	Define the current colour state of OpenGL (red, green, blue)
glMatrixMode(matrix)	Apply subsequent transformations to the specified matrix (GL_MODELVIEW or GL_PROJECTION)
glPushMatrix()	Store the current state of the currently selected matrix (modelview or projection) on the matrix stack
glPopMatrix	Recover the currently selected matrix by popping the top values from the matrix stack
glLoadIdentity()	Remove any existing transforms on the current matrix
gluOrtho2D(left, right, bottom, top)	Define a 2D orthographic projection with specified world coordinate viewport
gluLookAt(eyex,eyey,eyez, atx,aty,atz, upx,upy,upz)	Position the camera (*eye* coords), target it (*at* coords) and define its 'up' vector (*up* coords)
gluPerspective(fov, aspect, near, far)	Define a 3D perspective projection with field-of-view (fov), aspect ratio, near and far clipping planes
glTranslatef(x,y,z)	Translate the coordinate system of the current matrix
glRotatef(a,x,y,z)	Rotate the coordinate system of the current matrix
glScalef(x,y,z)	Scale the coordinate system of the current matrix along each of its principal axes
glutDisplayFunc(myfunction)	Register your display (render) function with glut
glutTimerFunc(delay, myfunction, data)	Register a function with glut for delayed execution – delay is in milliseconds